

In re Application of WANG et al.
Serial No. 09/447,501

REMARKS

The Office action has been carefully considered. The Office action rejected claims 1, 48, and 49 under 35 U.S.C. § 112, second paragraph for insufficient antecedent basis. Claims 1-16 and 27-47 were rejected under 35 U.S.C. § 103(a) as being unpatentable over Marsland, U.S. Patent No. 6,047,124 (hereinafter "Marsland") in view of Geist Jr., U.S. Patent No. 5,491,808 (hereinafter "Geist"). Claims 48-53 were rejected under 35 U.S.C. § 102(b) as being anticipated by Geist. Entry of the response and reconsideration of the claims under the provisions of 37 C.F.R. 1.116 is earnestly solicited.

By present amendment, claims 1, 4-7, 11-13, 16, 27, 29, 31, 32, 35, 40, 44-47, 48, and 49 have been amended. No other claims have been added or cancelled. Claims 1-17 and 27-53 remain pending. Applicants submit that the claims as presented were patentable over the prior art of record, and that the amendments herein are for purposes of clarifying the claims and/or for expediting allowance of the claims. Reconsideration is respectfully requested.

Applicants thank the Examiner for the interview held (by telephone) on October 23, 2003. During the interview, the Examiner and applicants' attorney discussed the claims with respect to the prior art. The essence of applicants' position is incorporated in the remarks below.

Turning to the 35 U.S.C. § 112, second paragraph rejections, the Office action rejected to claims 1, 48, and 49 for insufficient antecedent basis. Applicants have amended these claims in response. Reconsideration and withdrawal of the rejections of these claims under 35 U.S.C. § 112 is respectfully requested.

In re Application of WANG et al.
Serial No. 09/447,501

Turning to the § 103 rejections, the present invention relates to testing kernel mode components, such as drivers loaded by an operating system, while those components are operating. Tests that correspond to specific types of errors to be monitored are identified to the kernel at the time of loading the component. To test the driver, when the loaded kernel component makes a system call, the call is re-vectored to a special kernel mode verifier component that controls execution of the system calls, and establishes tests for the specific errors that were identified for that component. For example, if the type of error to be monitored was directed to memory-related errors, then the verifier component employs a series of memory monitoring techniques to watch for errors. Such memory monitoring techniques include isolating the memory allocated to a component to see if the component writes to memory locations beyond that which was allocated, testing memory usage, and marking deallocated memory. The verifier may also check whether all allocated memory is deallocated when the component is unloaded, to detect memory leaks. Additionally, the verifier may check whether any requested kernel resources were deleted when the component unloads. These resources may include queues, lookaside lists, worker threads, timers and other resources. To increase the chances of finding errors that only occur in certain conditions such as when available memory is low, the component verifier also may to simulate low resource conditions to discover improper component behavior at such times. Note that the above description is for informational purposes only, and should not be used to interpret the claims, which are discussed below.

Applicants have amended claim 1 to substantially include the limitations found in dependent claims 12 and 13. In particular, claim 1 generally recites a kernel mode driver

In re Application of WANG et al.
Serial No. 09/447,501

verify that is capable of testing the kernel mode driver by simulating a low resource condition including failing requests for memory pool allocation.

The Office action rejected claim 12 (and hence the subject matter of amended claim 1) on the same grounds as it rejected claim 6. The rejection of claim 6, however, states nothing about simulating a low resource condition as recited by claim 12 (and now claim 1). Applicants respectfully submit that neither Marsland nor Geist disclose or suggest anything regarding simulating a low resource condition. Indeed, the only references to the word "simulate" or its variants in Geist refer to simulating the instructions replaced by the thunk. Marsland does not even include the word simulate.

In order to establish *prima facie* obviousness of a claimed invention, all of the claim limitations must be taught or suggested by the prior art. *In re Royka*, 490 F.2d 981, 180 USPQ 580 (CCPA 1974). In addition, "all words in a claim must be considered in judging the patentability of that claim against the prior art." *In re Wilson*, 424 F.2d 1382, 1385, 165 USPQ 494, 496 (CCPA 1970). Further, if prior art, in any material respect teaches away from the claimed invention, the art cannot be used to support an obviousness rejection. *In re Geisler*, 116 F.3d 1465, 1471, 43 USPQ2d 1362, 1366 (Fed Cir. 1997). As the Office action has failed to show where all of the claim limitations are taught or suggested, the Office action has not established a *prima facie* case for obviousness. At least for these reasons, claims 1 and 12 and the claims that depend thereon are patentable over the cited references.

Turning to the rejection of claim 27, claim 27 generally recites receiving a request for an allocation of memory, determining a location of memory to allocate, restricting access to areas bounding the location, wherein an access request to at least one of the areas

In re Application of WANG et al.
Serial No. 09/447,501

results in an access violation, allocating the memory, and monitoring the areas for an access violation.

The Office action asserts that Geist teaches that a driver/program is "only allowed" to use memory allocated to that driver/program. Applicants strongly disagree. As is understood by those of ordinary skill in the art, while a program may be given a block of memory (e.g., through a malloc), there is nothing restricting the program (especially a kernel mode program) from accessing areas outside of that block. For example, a program may ask for a block of memory. In response, the program may be given a suitably sized block of memory. In programming languages that allow pointer access to memory (e.g., C, C++, assembly language, and so on), there is nothing that stops the program from attempting to access memory before or after the block of memory. These languages leave it to the programmer to make sure that the program will not access memory outside of the block of memory. If a programmer makes a mistake or miscalculation, the program or system can easily crash due to another program's memory being corrupted by an improper memory access.

As is also understood by those of ordinary skill in the art, there is a tradeoff for allowing pointer access to memory. Pointer access improves performance but also allows a corruption of memory as a program can access memory that is not assigned to it (e.g., by simply adjusting a pointer). The potential to corrupt or write in non-assigned memory is particularly true for kernel mode drivers that have no restrictions as to what memory they can access. With operating system components, such as kernel mode drivers, the speed with which the component executes often affects the performance of the entire system. For this reason, such drivers are typically programmed using a language that allows pointer

In re Application of WANG et al.
Serial No. 09/447,501

access to memory. Unfortunately, if a driver accesses memory outside of its allotted memory, the driver can easily cause the entire system to crash.

In any case, those of ordinary skill in the art readily recognize that when a program obtains memory via a malloc that the program *can* (even if it should not) access memory other than that allocated via the malloc. Hence, such a program is *not* “only allowed” to use the memory allocated to it as asserted in the Office action, but can access memory it is not supposed to access. Indeed, that is one common software bug that the present invention was invented to detect. Certainly, the Office action has not identified anything in the cited references that discloses, suggests, or remotely hints that “any access request to at least one of the areas [bounding the allocated location] results in an access violation” as recited in claim 27. Applicants respectfully submit that the cited references in any permissible combination contain no such disclosure or suggestion. At least for this reason, claim 27 and the claims that depend thereon are patentable over the cited references.

Dependent claim 29 recites “restricting access to deallocated memory space, wherein any access request to the deallocated memory space results in an access violation.” The Office action asserts without any support from a cited reference that “[i]t is inherent that all deallocated memories are inaccessible until it [sic] is allocated.” This assumes that a deallocated memory will not be accessed by a misbehaving program (e.g., one that attempts to access the memory before it has been allocated, one that attempts to access the memory after it has been deallocated, and so on). As pointed out above, even though a program should not access memory not allocated to it, through a programmer’s mistake, miscalculation, or intentional action (e.g., a virus), a program *can access* such memory before allocation or after deallocation. Indeed, that is another common software bug that

In re Application of WANG et al.
Serial No. 09/447,501

the present invention was invented to detect. This is particularly true for programs that have elevated access privileges (e.g. kernel mode drivers). The Office action has not pointed to anything in the references, whether considered alone or in any permissible combination, that discloses or suggests the subject matter of claim 29. At least for this additional reason, claim 29 and the claims that depend thereon are patentable over the cited references.

Turning to the rejection of claim 32, the Office action cites Geist, column 11, lines 12-22 to allege that Geist teaches the step of generating an error that indicates that resources remain associated with a driver that has unloaded. In particular, the Office action indicates that the "freed NULL" of Geist indicates this type of error. Applicants strongly disagree. The Office action appears to have misinterpreted the meaning of a freed NULL.

A freed NULL occurs when a program frees a pointer that points to nothing. Typically, a pointer is initialized to NULL when a request for memory is denied (e.g., for insufficient memory). For example if after the statement,

```
pointer = (char *) malloc(100);
```

the variable pointer has a value of NULL, this indicates that no memory was allocated by malloc for the pointer to point to. If a program does not check for and take appropriate action for this condition, e.g.:

```
if ((pointer = (char *) malloc(100)) == NULL)
{
    printf("We are out of memory")
    exit(NO_MEMORY)
}
```

the program might later attempt to access non-allocated memory using the pointer. This is an example of how a programmer can make a mistake or miscalculation (e.g., that memory

In re Application of WANG et al.
Serial No. 09/447,501

will always be returned), as previously discussed. Freeing a NULL pointer does not indicate that "resources remain associated with the driver" as recited in claim 32. It simply means that a program is attempting to free memory that was never allocated. Thus, the error "freed NULL" indicates nothing regarding whether a driver still has resources remaining after it has unloaded. Rather, if anything, it means that a program is attempting to free memory that was never allocated.

Similarly, the other errors mentioned in Geist column 11, lines 17-22, neither disclose nor suggest anything regarding "if resources remain associated with the driver, generating an error" as recited in claim 32. At least for this reason, claim 32 and the claims that depend thereon are patentable over the cited references.

Turning to the rejection of claim 31, claim 31 generally recites generating an error *at the time* a driver should not have memory allocated to it. As discussed previously, the cited references do not disclose generating an error if a resource such as memory remains allocated when it should not be. Certainly they do not disclose, suggest, or remotely hint at generating an error *at the time* the driver should not have memory allocated to it. At least for this reason, claim 31 is patentable over the cited references.

Turning to the rejection of claim 35, the Office action alleges that claim 1 meets the limitations of claim 35 "except for an operating system that includes an interface." Applicants disagree that claim 35 contain similar limitations as claims 1 and respectfully submit that each claim should be interpreted based on its own claim language and limitations. Furthermore, the rejection of claim 1 never addressed "allocating memory *for the driver to use* from a pool of memory *other than* a memory pool normally allocated from when the driver is operating unmonitored" as recited in claim 35. The memory allocated to

In re Application of WANG et al.
Serial No. 09/447,501

the driver is not just memory that is used to track the accesses of the driver; rather, it is memory *for the driver to use*. Furthermore, it is allocated from a pool of memory other than the memory pool normally allocated from when the driver is operating unmonitored. Neither Marsland nor Geist in any permissible combination disclose, suggest, or hint at this subject matter as recited in claim 35. At least for this reason, claim 35 and the claims that depend thereon are patentable over the cited references.

Turning to claim 38, Applicants note that the Office action does not contain any reasons for rejecting dependent claim 38. Applicants respectfully submit that claim 38 is not disclose or suggested by a permissible combination of the cited references. At least for this reason, claim 38 is patentable over the cited references.

As argued in a response dated April 21, 2003, and incorporated in this response, Applicants maintain that the Marsland and Geist cannot be combined in the manner suggested by the Office action as this is taught away from in the references. At least for this reason, claims 1-16 and 27-47 are patentable over the cited references.

Turning now to the 35 U.S.C. § 102(b) rejections of claims 48-53, claim 48 recites selecting one or more tests for verifying functionality of a component, modifying a request for system services to include execution of the selected tests, wherein one of the tests includes restricting access to a resource, such that an attempted access to the resource causes an access violation, executing the modified test, and generating errors for any test failures.

Nothing in the Geist or the cited references discloses, suggests, or remotely hints at having *a test* that restricts access to a resource such that an attempted access to the resource

In re Application of WANG et al.
Serial No. 09/447,501

causes an access violation. At least for this reason, claim 48 and the claims that depend thereon are patentable over the cited references.

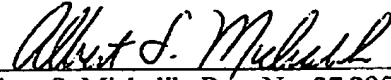
In re Application of WANG et al.
Serial No. 09/447,501

CONCLUSION

In view of the foregoing remarks, it is respectfully submitted that claims 1-17 and 27-53 are patentable over the prior art of record, and that the application is good and proper form for allowance. A favorable action on the part of the Examiner is earnestly solicited.

If in the opinion of the Examiner a telephone conference would expedite the prosecution of the subject application, the Examiner is invited to call the undersigned attorney at (425) 836-3030.

Respectfully submitted,



Albert S. Michalik, Reg. No. 37,395
Attorney for Applicant
Law Offices of Albert S. Michalik, PLLC
704 - 228th Avenue NE, Suite 193
Sammamish, WA 98074
(425) 836-3030
(425) 836-8957 (facsimile)